Feisty: Modifikasi Block Cipher AES dengan Jaringan Feistel

Arieza Nadya – 13512017
Program Sarjana Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13512017@std.stei.itb.ac.id

Junita Sinambela – 13512023
Program Sarjana Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13512023@std.stei.itb.ac.id

***Abstrak*—Makalah ini berisi analisis algoritma *cipher* baru menggunakan modifikasi berdasarkan algoritma AES dan *feistel network*. AES merupakan *block cipher* yang telah banyak digunakan dan telah diakui NSA untuk dapat dipakai mengenkripsi informasi rahasia dengan level SECRET. Algoritma hasil modifikasi menggunakan jaringan feistel sebagai algoritma utamanya namun pembangkitan kunci dilakukan dengan menggunakan AES. Algoritma Feisty memakai blok dengan panjang 256 bit dan menggunakan kunci 128 bit, sehingga dari sisi keamanan, panjang kunci aman dari *brute force*.**

Kata kunci—block cipher, jaringan Feistel, Advanced Encryption Standard, Rijndael.

I. PENDAHULUAN

Block cipher merupakan algoritma enkripsi kunci simetrik yang cara kerjanya per-*block*, yaitu pada *bitstrings* yang panjangnya tetap [3]. Kebanyakan dari *block cipher* merupakan bagian dari *iterated block cipher* yang berarti blok dari *plaintext* diaplikasikan sebuah atau lebih transformasi sehingga menghasilkan *ciphertext* dengan panjang blok yang sama, dan proses tersebut diulang pada setiap *round* [4]. *Block cipher* memiliki dua algoritma yang berpasangan untuk masing-masing enkripsi dan dekripsi [5]. Algoritma *block cipher* yang sering dipakai meliputi DES (*Data Encryption Standard*), yang menggunakan *Feistel Network*, dan AES (*Advanced Encryption Standard*) [3], yang menggunakan *substitution-permutation network* [6]. Claude Shannon mengemukakan dua aspek keamanan operasi pada kriptografi, yaitu *confusion* dan *difusion*. *Feistel Network* yang dipakai pada algoritma DES menggunakan operasi-operasi yang dilakukan berulang untuk membuat fungsi yang menghasilkan *confusion* dan *diffusion*. Analisis oleh Michael Luby dan Charles Rackoff membuktikan bahwa pada konstruksi dari *feistel network* jika *round function*-nya merupakan fungsi pseudorandom yang aman secara kriptografis, dengan K_i digunakan sebagai *seed*, maka *block cipher* tersebut cukup hanya menggunakan 3 *round* untuk menghasilkan permutasi pseudorandom, dan 4 *round* untuk menghasilkan permutasi pseudorandom yang kuat [7].

II. TEORI SINGKAT

A. Algoritma AES

AES (*Advanced Encryption Standard*) merupakan standar algoritma enkripsi data elektronik menggunakan algoritma Rijndael yang dipublikasikan oleh NIST. Algoritma ini dibuat dan diajukan oleh Vincent Rijmen dan Joan Daemen. Algoritma AES didesain atas dasar *code simplicity*, ketahanan atas semua *known attacks*, kecepatan dan *compactness* dari kode untuk bermacam-macam *platform* [1].

Algoritma Rijndael menggunakan 10 *round* dengan operasi yang sama pada setiap *round*-nya, kecuali untuk *round* terakhir. Operasi yang digunakan pada algoritma ini terdiri dari 4 transformasi yang diaplikasikan terhadap *state bytes* yang ditulis dalam bentuk hexadecimal. Transformasi yang dilakukan adalah sebagai berikut secara terurut [2]:

1. Transformasi *Sub Bytes*

   Substitusi masing-masing *byte* pada *state bytes* dengan menggunakan S-box (*substitution box*).

2. Transformasi *Shift Rows*

   Pada transformasi ini *bytes* pada 3 baris terakhir pada *state bytes* (untuk AES-128) di-*shift* secara siklik. Nilai *shift* ditentukan oleh nomor baris.

3. Transformasi *Mix Columns*

   Transformasi ini menjadikan setiap kolom sebagai bilangan polinomial 4 suku yang dikalikan dengan polinom $a(x)$ mod $(x^4 - 1)$. $a(x)$ tetap yaitu

   $$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{01\}$$

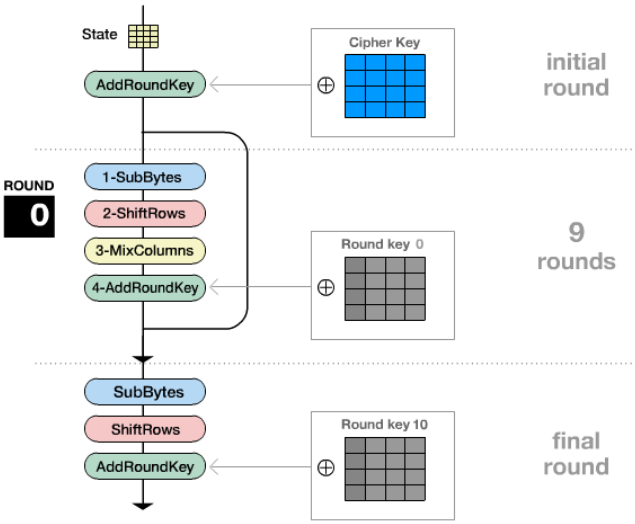
   Pada *round* terakhir bagian transformasi *mix columns* dihilangkan.

4. Transformasi *Add Round Key*

   Sebuah *round key* yang berbentuk matriks 4x4 berisi elemen *bytes* ditambahkan ke *state bytes* dengan operasi XOR. Penambahan *round key* dilakukan dari pada saat *round* 0 sebelum transformasi-transformasi lain dilakukan.

Untuk melakukan dekripsi *cipher* yang menggunakan algoritma Rijndael hanya perlu melakukan kembali *round-round* tersebut secara terbalik.

Pemilihan jumlah *round* 10 untuk AES-128 (12 untuk AES-192 dan 14 untuk AES-256) didasarkan pada percobaan *shortcut attack* terhadap algoritma ini. Tidak ditemukan *shortcut attack* yang berhasil untuk jumlah *round* di atas 6, namun ditambahkan 4 *round* sebagai *security margin* [1].



Gambar 1 Algoritma Advanced Encryption Standard *(AES)* [8]

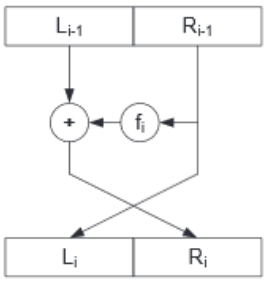
Pada *Gambar* ***1*** dapat dilihat bagaimana fungsi dalam AES diurutkan.

B. *Feistel Network*

Feistel network merupakan suatu konstruksi yang digunakan untuk membuat sebuah algoritma enkripsi simetrik menggunakan konsep *bit shuffling* dan substitusi *byte* untuk membuat *confusion* dan *diffusion*, dua aspek *cipher* aman yang dikemukakan oleh Claude Shannon (1945). Konsep jaringan feistel ini pertama dikemukakan oleh Horst Feistel saat mendeskripsikan *cipher Lucifer*, yang revisinya menjadi algoritma DES. *Feistel network* biasanya bekerja dalam *d rounds*, biasanya sekitar 12 – 16 *rounds*. Pada *round* ke-*i* berikut adalah operasi yang dilakukan pada blok yang dikenai *feistel network* [3]:

1. Input dibagi menjadi dua bagian yaitu L_{i-1} dan R_{i-1},
2. Bagian kanan input, R_{i-1}, dikenai fungsi *round f*, sehingga menghasilkan $f(R_{i-1})$,
3. Nilai dari hasil *round function* sebelumnya di-XOR-kan dengan nilai dari bagian kiri input menghasilkan $L_{i-1} \oplus f(R_{i-1})$,
4. Nilai dari bagian kanan dan kiri dipertukarkan sehingga menghasilkan output $L_i \| R_i := R_{i-1} \| L_{i-1} \oplus f(R_{i-1})$.

Pada Gambar 2 dapat dilihat bagaimana fungsi dalam AES diurutkan.



Gambar 2 Ilustrasi 1 round feistel network [7]

III. RANCANGAN ALGORITMA

Algoritma Feisty menggunakan jaringan Feistel sebagai algoritma utamanya. Setiap blok berukuran 256 bit, dengan kunci berukuran 128 bit. Kunci ini nantinya akan digunakan sebagai parameter untuk fungsi di dalam jaringan feistel.

Feistel diiterasi sebanyak 8 kali, dengan parameter yang berbeda untuk fungsi dalam setiap putarannya. Adapun kunci dibangkitkan dengan menggunakan AES hasil modifikasi.

A. *Modifikasi AES*

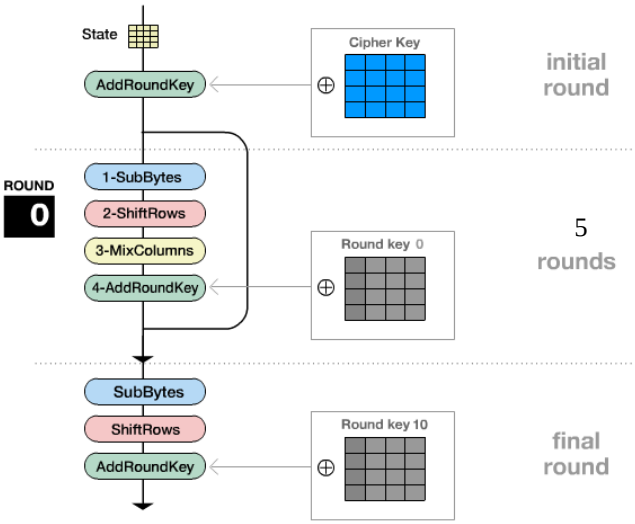
AES yang dimodifikasi menggunakan keseluruhan fungsi utamanya, yaitu: *sub bytes*, *shift rows*, *mix columns*, dan *add round key*. Keempat fungsi ini tidak diubah sama sekali, dan hanya mode enkripsinya saja yang digunakan, sehingga algoritma ini mengandung operasi substitusi dan transposisi.

Ukuran *plain text* yang digunakan adalah 128 *bit* (16 *bytes*), sama dengan ukuran normal kunci AES. Hal ini disebabkan masing-masing ukuran blok yang akan menggunakan AES hasil modifikasi ini berukuran 128 bit.

Selain itu, berdasarkan hasil *shortcut attack* yang diuji pada AES, AES hasil modifikasi hanya akan menggunakan 6 kali putaran. Hal ini didasarkan atas pertinbangan agar kunci tidak dapat diprediksi oleh penyerang.

Dari kesimpulan tersebut, maka modifikasi AES pada algoritma ini menggunakan AES dengan putaran (*round*) sebanyak 6 kali.

Hasil modifikasi AES dapat dilihat pada Gambar 3.*Gambar* ***1***



Gambar 3 Hasil modifikasi algoritma Advanced Encryption Standard *(AES)* [8]

B. *Pembangkitan Kunci pada Jaringan Feistel*

Pembangkitan kunci untuk setiap iterasi pada putaran jaringan Feistel dilakukan dengan cara membagi kunci menjadi dua blok yang masing-masing berukuran 128 bit. Blok putaran pertama adalah 128-bit pertama kunci, sedangkan blok putaran kedua adalah 128-bit terakhir kunci. Kunci putaran pertama adalah blok pertama hasil pembagian kunci, sedangkan kunci putaran kedua menggunakan blok kedua pembagian kunci. Kedua kunci tersebut kemudian dienkripsi menggunakan AES hasil modifikasi dengan blok pertama sebagai *plain text* dan blok kedua sebagai kunci, menghasilkan sebuah *blok cipher* yang nantinya akan dimasukkan sebagai kunci putaran ketiga. Kunci putaran keempat menggunakan hasil enkripsi menggunakan kunci kedua sebagai *plain text* dan kunci ketiga sebagai kunci enkripsi. Begitu seterusnya sampai putaran ke-8.

Kunci round 1 ← 128-bit terkiri kunci
Kunci round 2 ← 128-bit terkanan kunci
Kunci round 3 ← AES(Kunci round 1, Kunci round 2)
Kunci round 4 ← AES(Kunci round 2, Kunci round 3)
Kunci round 5 ← AES(Kunci round 3, Kunci round 4)
Kunci round 6 ← AES(Kunci round 4, Kunci round 5)
Kunci round 7 ← AES(Kunci round 5, Kunci round 6)
Kunci round 8 ← AES(Kunci round 6, Kunci round 7)

Gambar 4 Pembangkitan kunci untuk setiap putaran pada jaringan Feistel

Kedelapan kunci ini digunakan dengan memperhatikan urutan putaran yang akan menggunakan kunci sebagai salah satu parameternya. Karena ada ketergantungan antar kunci dimulai pada kunci ketiga, maka dalam melakukan dekripsi, semua kunci harus dibangkitkan terlebih dahulu untuk mendapatkan kunci terakhir (kunci kedelapan).

C. *Fungsi pada Jaringan Feistel*

Jaringan feistel menggunakan fungsi yang dapat diubah sesuai dengan keinginan perancang algoritma kriptografi yang akan menggunakannya. Algoritma ini menggunakan AES hasil modifikasi untuk melakukan enkripsi bagian kanan blok yang akan dienkripsi untuk kemudian di-XOR dengan bagian kiri blok. Bagian kanan blok yang menjadi masukan AES akan menjadi bagian *plain text* yang akan dienkripsi menggunakan kunci putaran.

D. *Hasil Pengujian*

Dari algoritma di atas, maka dapat dilihat bahwa kunci untuk masing-masing putaran pada jaringan Feistel bergantung pada kunci awal yang dimasukkan oleh pengguna.

Plain text:
6bc1bee22e409f96e93d7e117393172a ae2d8a571e03ac9c9eb76fac45af8e51
Key:
2b7e151628aed2a6abf7158809cf4f3c 6bc1bee22e409f96e93d7e117393172a
Cipher text:
4d7919c74bef8ca4c7bc975c9143af68 92533b6cdf8b2072b6c4e4a0f05acdd4

Gambar 5 Hasil pengujian pertama algoritma Feisty menggunakan mode ECB

Plain text:
30c81c46a35ce411e5fbc1191a0a52ef f69f2445df4f9b17ad2b417be66c3710
Key:
2b7e151628aed2a6abf7158809cf4f3c 6bc1bee22e409f96e93d7e117393172a
Cipher text:
a549d4f269ab1ac5122a619d6a7eb10e 6bbc3f405cc18b8122d83d3ec8038ca1

Gambar 6 Hasil pengujian kedua algoritma Feisty menggunakan mode ECB

Pada Gambar ***5*** dan Gambar ***6*** dapat dilihat bahwa hasil enkripsi kedua *plain text* menghasilkan *cipher text* yang berbeda-beda. Kedua *cipher text* di atas juga dapat didekripsi dengan baik untuk fungsi yang sama.

IV. Analisis Keamanan

Panjangnya kunci adalah 256 bit, sehingga kemungkinan kunci ada sebanyak 2^{256} kemungkinan, yaitu sekitar 1.158 x 10^{77}. Sehingga, untuk melakukan *brute force* untuk melakukan

dekripsi *cipher text*, dengan *processor* yang memiliki kemampuan untuk mencoba satu kemungkinan kunci selama 1 ms, maka dibutuhkan waktu yang sangat lama untuk menemukan kuncinya.

Karena algoritma ini menggunakan AES yang jumlah putarannya dikurangi dari jumlah putaran asalnya, sehingga algoritma ini memiliki efek subsstitusi dan transposisi.

Selain itu, karena adanya ketergantungan kunci suatu putaran jaringan Feistel ke dua kunci pada putaran sebelumnya (kecuali kunci pada putaran pertama dan putaran kedua), maka algoritma memiliki efek difusi. Jaringan Feistel juga menyebabkan adanya efek difusi pada hasil enkripsi setiap bloknya.

V. KESIMPULAN

Dari perancangan dan implementasi algoritma Feisty, didapatkan kesimpulan, yaitu:

1. algoritma telah memenuhi mengacu kepada prinsip *diffusion* dan *confusion* dari Shannon,
2. algoritma ini dapat diterapkan dalam mode ECB, CBC, maupun CFB.

Algoritma ini sendiri menggunakan fungsi-fungsi pada AES tanpa ada modifikasi untuk setiap fungsinya. Namun, kompleksitas pada AES sendiri menyebabkan fungsi yang diimplementasikan menjadi sulit untuk dianalisis.

REFERENCES

[1] J. Daemen and V. Rijmen, AES proposal: Rijndael (1999).

[2] *Advanced Encryption Standard*, FIPS PUB 197, 2001.

[3] M. Backes. (2011). *Block Ciphers* [Online]. Available: http://web.cs.du.edu/~ramki/courses/security/2011Winter/notes/feistelProof.pdf.

[4] *Junod, Pascal & Canteaut, Anne (2011). Advanced Linear Cryptanalysis of Block and Stream Ciphers. IOS Press. p. 2. ISBN 9781607508441.*

[5] *Cusick, Thomas W. & Stanica, Pantelimon (2009). Cryptographic Boolean functions and applications. Academic Press. pp. 158–159. ISBN 9780123748904.*

[6] *van Tilborg, Henk C. A.; Jajodia, Sushil, eds. (2011). Encyclopedia of Cryptography and Security. Springer. ISBN 978-1-4419-5905-8.*, p. 1268.

[7] *Luby, Michael; Rackoff, Charles (April 1988), "How to Construct Pseudorandom Permutations from Pseudorandom Functions", SIAM Journal on Computing* ***17*** *(2): 373–386, doi:10.1137/0217022, ISSN 0097-5397*

[8] Rijndael Animation [Online]. Available: http://www.formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng.swf